

APPLICATION  
FOR  
UNITED STATES LETTERS PATENT

TITLE: HOST DOWNLOADED MULTI-SEGMENT DSP CODE  
FILE FORMAT

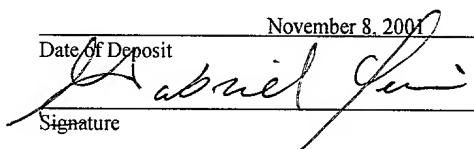
APPLICANT: THOMAS A. SCHULTZ

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL58602229US

I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C 20231.

Date of Deposit November 8, 2001

Signature 

Gabe Lewis  
Typed or Printed Name of Person Signing Certificate

100069001 200590001

## **HOST DOWNLOADED MULTI-SEGMENT DSP CODE**

### **FILE FORMAT**

#### **TECHNICAL FIELD**

[0001] This invention relates to device drivers for computer systems, and more particularly to device drivers for downloading operating code to a processor.

#### **BACKGROUND**

[0002] Computer systems typically include one or more device drivers to allow the operating system of the host system to support certain hardware devices such as a modem, network interface card, printer, and many others. The hardware device may include a processor and local memory to implement the device functions. The device driver may include code that communicates directly to the hardware device and its local memory, to control the operation of the hardware device processor. To reduce the cost of the hardware device, the amount of local memory in a device is typically minimized. However, the amount of local memory generally cannot be reduced to less than what is required for storing the downloaded instructions from the device driver. Therefore, it is desirable to reduce the quantity of downloaded instructions that are stored in the local memory of a hardware device.

### DESCRIPTION OF DRAWINGS

[0003] FIG. 1 is a block diagram of a driver coupled to a DSP.

[0004] FIG. 2 is a diagram of a jump table.

[0005] FIG. 3 is a flow diagram of a device driver download to a processor.

[0006] Like reference symbols in the various drawings indicate like elements.

### DETAILED DESCRIPTION

[0007] Figure 1 shows a device driver 10 for downloading multi-segment code to a digital signal processor (DSP) 12 upon request. The DSP 12 includes a limited amount of local memory 13 in which downloaded code segments may be stored. The device driver 10 includes a code file 14 that is loaded into host memory. The code file 14 contains firmware code 16 for controlling the DSP 12. The firmware code 16 includes separate code segments 22 with corresponding segment code addresses 24 that may be used to locate the code segments 22 in memory. Each of the code segments 22 may include one or more instruction codes for controlling the operation of the DSP 12 during different operating modes. For example, when the DSP 12 is switched from a sensing mode of operation to an outputting mode, a code segment 22 that includes the operating instructions for

the outputting mode may be requested by the DSP 12 from the driver 10. By switching in code segments only when needed, the DSP 12 minimizes the quantity of local memory 13 that is required, thereby reducing the cost of the interface assembly.

**[0008]** The code file 14 may also include a description 18 of the firmware code 16 as well as build information 20. The code description 18 is in human readable format to enhance understanding of the operation of the firmware code 16. The build information 20 is also human readable to improve maintenance and upgradability of the code file 14.

**[0009]** A jump table 26 maps segment labels to the address locations of the code segments 22 through a mechanism such as an array of memory pointers. The segment labels may be any combination of characters such as a numeric or alpha-numeric. Each segment label identifies an absolute address corresponding to a code segment 22. By mapping segment labels to the absolute address of the code segments 22, the DSP 12 can request a code segment 22 without knowledge of the absolute address at which the code segment 22 is located. The address location of a code segment 22 may be described by the starting address of the code segment 22, the ending address of the code segment 22 in combination with the segment size, and all other means of identifying the absolute address of the code segment 22.

[0010] Figure 2 shows the formation of the jump table 26. Initially, the jump table 26a is created with segment labels 28 that are mapped to the relative addresses 30 of the code segments 22. Each of the code segment relative addresses 30 describes the location of a code segment 22 in relation to the other code segments 22. When the absolute addresses 32 of the code segments 22 are determined, the relative addresses 30 may be fixed-up to reflect the absolute addresses 32 for the code segments 22. The absolute addresses 32 may be determined once a location in memory has been allocated for the code file 14, or after the code file 14 is loaded into memory by using a conversion factor. For example, if the code file 14 is sequentially loaded into memory, the address offset of the code file 14 in memory may be used as the conversion factor for fixing-up the relative addresses. The offset may be used as a constant by which each of the relative addresses 30 is incremented so that the jump table 26b includes the absolute addresses 32 of the code segments 22. By overwriting the relative addresses 30 with the absolute addresses 32, the computation speed of the driver 10 is increased and the amount of memory used is reduced. Instead of overwriting the relative addresses 30, the offset may be stored and then used in combination with the relative addresses 30 to compute the absolute addresses 32 when a code segment 22 is downloaded.

Also, the relative addresses 30 may be retained in addition to the absolute addresses 32.

**[0011]** A loader 34 loads the code file 14 and jump table 26 into memory. The loader 34 may also receive and respond to download requests from the DSP 12 for code segments 22. The download requests identify the requested code segment 22 by a corresponding segment label 28. In response, the loader 34 locates the requested code segment 22, and then sends the code segment 22 to the DSP 12.

**[0012]** Figure 3 shows a process for downloading a code segment 22 to the DSP 12. Beginning at state 40, the driver 10 is loaded into the memory of a host computer system. A form of bootstrapping operation may be used to load the device driver 10 into memory. First, the loader 34 is loaded into memory. Then, the loader 34 loads the code file 14 and jump table 26 into the host memory. The loader 34 also supports dynamic download of a revised code file 14. For example, if the code file 14 and jump table 26 have been previously loaded into host memory, the loader 34 may load a revised code file 14 and jump table 26 into host memory so that a dynamic download to the DSP 12 may be performed without resetting the host computer. Continuing on to state 42, the loader 34 determines the memory location of the loaded code file 14. At state 44, the conversion factor is determined for generating absolute addresses 32 corresponding to

the relative addresses 30 in the jump table 26. The conversion factor is used to fix-up the relative addresses 30 in the jump table 26 so that absolute address references 32 are created.

[0013] Continuing on to states 46 to 50, a dynamic download operating phase of the driver 10 is shown. Once the driver 10 is loaded into the host system memory (not shown), the driver is ready to download code segments 22 to the DSP 12. At state 46, a download request from the DSP 12 is received. The download request identifies a requested code segment 22 by a corresponding segment label 28. Continuing on to state 48, the jump table 26b is queried to determine the segment code address 32 of the identified code segment 22. At state 50, the code segment 22 is then downloaded from the referenced location in the host memory to the DSP 12.

[0014] A number of embodiments of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. Accordingly, other embodiments are within the scope of the following claims.